

\$2.50

October 1976 Volume 4 Number 10

CONCERNING FILE MANAGEMENT SYSTEMS by Fred Gruenberger

File Management Systems were introduced in the mid-60's and constitute today perhaps the leading example of packaged proprietary programs. This article attempts to cover the following aspects of this significant advance in business data processing:

1. Just what is a File Management System?
2. What are its capabilities?
3. What is the proper role of a File Management System--who should use one, and for what?
4. Why has the spread of such systems been so slow?

The illustrative example to be used is the Mark IV system marketed by Informatics Inc. Software Systems Company, partly because of the author's familiarity with Mark IV and partly because Mark IV is pre-eminent in the field.

The business of business is the processing of files. A file is a collection of related records. A record is a string of characters made up of fields. The simplest file has fields of fixed length and fixed position (naturally derived from punched card files); for example, the telephone book:

LAST NAME	FIRST NAME	ADDRESS	NUMBER
20	10	30	8
JONES	JOHN	1234 Main St., Reseda	345-6789

For various practical reasons, all sorts of variations on this simple format are possible:

1. Fields may have variable length. Last names, for example, may vary from 1 letter to over 20. If the fields have variable length, then each field must include its length, or the record must contain an index of pointers to the starting character of each field.
2. There may be a variable number of fields for a given category. In a payroll file, for example, if the names of dependents are to be included, there could be from none to, say, 20 such sub-fields.
3. If stored on a direct access device, each record may contain one or more pointers to the physical address of its successor. Thus, the telephone file might be ordered in three ways:

Please turn to page 5

Log 43 1.633468455579586526405088153229222158808774884380093
 ln 43 3.761200115693562423472842513345847035559136184881555
 $\sqrt{43}$ 6.557438524302000652344109997636001627926966319883790
 $\sqrt{43}$ 3.503398060386724170614333758189129737248756786915392
 $\sqrt{43}$ 1.456621934766155391441781950828466209853127201851841
 $\sqrt{43}$ 1.038328284538685964669582594489982111422294037314318
 e^{43} 4727839468229346561.474457562744280370819751962380938
 170967197960251507498949982309854
 π^{43} 2384759161287667022584.943464869338170313079932281203
 605661148818693252220271615517
 $\tan^{-1} 43$ 1.547544703984433703765045019098361071107979797950989

All subscription prices are for remittance in U.S. dollars. A charge of \$3 is made for billing. Basic subscription price is \$16 for one year (\$30 for two years) plus the following for postage and order processing:

U.S. \$1.50 per year

Canada and Mexico \$3 per year.

All other countries \$5 per year.

Back issues, when available, are \$2.50 each, or \$2 each for two or more (same or different issues).

Copies not received will be replaced free if we are notified within 90 days (U.S.) or 120 days (overseas).

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is \$20.50 per year, or \$17.50 if remittance accompanies the order. For Canada and Mexico, add \$1.50 to the above rates. For all other countries, add \$3.50 per year to the above rates. Back issues \$2.50 each. Copyright 1976 by POPULAR COMPUTING.

Editor: Audrey Gruenberger

Publisher: Fred Gruenberger

Associate Editors: David Babcock
Irwin Greenwald

Contributing editors: Richard Andree

William C. McGee

Thomas R. Parkin

Advertising Manager: Ken W. Sims

Art Director: John G. Scott

Business Manager: Ben Moore

Reproduction by any means is prohibited by law and is unfair to other subscribers.

- a. Alphabetically by last name.
- b. Numerically by telephone number.
- c. Alpha-numerically by address, and, to avoid having each record stored three times, each record would contain three pointers to the next in sequence on the three orderings.

4. Any given field in a record can be expressed in binary, in BCD, in packed decimal, in pure alpha, or in alpha-numeric format.

Other variations are possible, but in any event, files of this sort constitute the heart of any businesses data processing activity. What does one do with files?

LIST A

1. Create a file. This involves giving the file a name and defining one or more keys and the format of all fields in its records. A key is a field on which the file is ordered or accessed, such as the name field in the telephone file.

2. Update a file. New records are to be added; dead records are to be deleted; fields in specific records are to be changed or replaced; fields are to be modified. Additions or deletions of whole records may be made by operating only on the key; all other changes to the file require extensive edit checking to be sure that the proposed change agrees with the original definition of the fields involved. An inadvertent requested change of "add 3786 to the last name of record JONES, JOHN" can introduce chaos to a file.

3. Merge two files.

4. Search a file. For example, find all records for which last name exceeds 18 characters; find all records for which last name begins with S and the telephone exchange (i.e., the first 3 digits of the number) is 345.

5. Sort a file. For example, reorder the telephone file by last four digits of the number. This operation may produce a new file, which is a reordered copy of the old file. Usually, sorting is done to order input records, or for producing reports.

6. Clean up a file. Let's go back to the 7-card deck problem which appeared in our July issue. The problem was stated as follows:

A deck of cards bears six 3-digit numbers on each card, in columns 1 through 18. It is formed of a number of seven-card sub-decks; each sub-deck is identified by the 3-digit number in columns 1-3 of the first of the seven cards.

As the cards are read, a total is to be formed of the other 41 numbers in each sub-deck. After each sub-deck has been handled, its identifying number and the sum for that deck is to be printed.

Assume that a subroutine is available that will read a card and place its six numbers in words addressed at G, G+1, G+2, G+3, G+4, and G+5. The identifying numbers for the sub-decks are all over 500; all data numbers are less than 500. The end of the full deck is signalled by a sentinel card bearing the number 999 in its first three columns.

Any experienced person would howl at the assigned problem. Deck identification mixed in with the data? No cards identified within the sub-decks? The same card field used sometimes for data and sometimes for ID? The problem involved has not been analyzed properly, and the suggested format (devised by the customer of the DP center) just won't do. Now, the customer may have already keypunched great masses of data. Considering that data as a file, it should be cleaned up and reformatted, so that subsequent daily data will be in decent form for processing. Notice that each day's data (say, 300 7-day decks) can now be considered as a file to be updated.

7. Write reports from an updated file. The production of reports is the number one task of any installation; it is what management pays the installation to do. All other activities lead to the production of reports. Certain elements are common to all reports:

- a. The report should be clearly identified; that is, labelled.
- b. It should be dated.
- c. If it runs more than a single page, the pages should be numbered.
- d. If the report is classified, the classification must appear on every page, centered on the top and bottom lines of the page.
- e. All data items on the report should be formatted according to accepted standards: e.g., left zero suppression; fixed point (even if the calculations have been done in scientific notation); and so on.

Every item in List A is canonical; that is, programs to do any one of the tasks will all look identical--only the parameters change. Consider two programs that each merge two files; what are the possible parameters?

1. The key or keys on which the merge is to take place must be specified (and these need not be in the same place in the two input files).
2. Ascending or descending order on those keys?
3. Run controls; that is, what fields should be hash totalled; where are the input control numbers in each file; how many out-of-sequence errors will be tolerated before the merge is killed; is an "equals" condition an error or not--things like that.
4. What are the available resources for this merge? How much core is available; how much auxiliary storage is available?
5. What are the blocking factors (on tape)? File A could be blocked by 5's; file B blocked by 10's; but the merged file is to be blocked by 2's.
6. Is there a restart procedure, or must this merge run to completion to be run at all?

As long ago as 1957, programs were being written to generalize all the tasks listed above, so that none of them would ever need to be re-programmed. Such programs were the original generators, and they were written for report writing, sorting, merging, and file maintenance. A File Management System brings them all together in one large program whose input is normally only check marks on preprinted forms to indicate what is to be done to what files. In addition to performing the tasks requested of it, the system automatically furnishes extensive editing and default options. The editing features save the user from erroneous transactions that he would later regret. For example, a data field that has been defined to be unsigned numeric decimal has an attempted update with an overpunched card--the transaction will be rejected (with suitable error messages). The File Management System is simply not responsive to the cry "But my data is clean." If it is clean, it will process, and if it isn't clean, the user wants to know (or should want to know, which may be a different thing).

The default options take care of all the obvious things that should never have to be programmed again: heading lines on reports are centered; page numbers begin with 1 and are printed in the upper right corner of the report; the date is furnished by the system to every task it performs each day; 60 lines are printed on each 11-inch page; and so on. Each such default option can be overridden by the user, of course.

Or, consider matching keys between records in an updating operation. The "hard create" option dictates:

- If a record with this key is there, reject the transaction.
- If a record with this key is not there, create a record.

While the "soft create" option dictates:

- If a record with this key is there, update it.
- If a record with this key is not there, create one.

The list of these options is quite long; it is designed to take care of all the normal things to be done--no code needs to be written for that which is normally done. The system is also loaded with interlocks. For example, an inadvertent attempt to divide by zero is flagged; the result field is outputted to reports with a specific indicator character or is unaffected if it is a field being updated in the file.

A text processing feature allows for data manipulation of types like the following:

Replace all occurrences of MRS or MISS by MS and adjust field widths accordingly.

Replace all zip codes of 12345 with 12346.

Count all appearances of zip code 90XXX.

A File Management System encourages users to ask for what they want, which is exactly what they will get, without worrying about comparing A to B and where to branch on the various outcomes.

The Mark IV system interfaces with the 360/370 operating systems and others with the same architecture. Since the Mark IV program itself is under the control of the Informatics people, it is rigidly standardized. Mark IV programs of 1968 will run under current versions of OS--some 15 generations later.

The Mark IV system uses automated test methods so that its new releases pass all previous release tests. The system is marketed and serviced exactly the same way that hardware is. It must perform as specified in the manual, or it will be made to, and promptly. It is in the very nature of a File Management System that a user will eventually have his entire operation dependent on having it work accurately, reliably, and continuously, just as he depends on the hardware.

Which brings us to the point of this article. A File Management System does the job, and does it better than newly written programs could do. It surely pays for itself, as evidenced by the fact that several major corporations have bought the package many times each. Why, then, is the product so difficult to sell? There are probably on the order of 10,000 installations in the United States that could make profitable use of it, but only some 700 installations have been made in its eight years of existence. One would think that there would be no need to "sell" it any more. Some of the reasons for sales resistance may be the following:

1. The extra checking features--designed to prevent any errors in transaction processing--obviously cost storage space and CPU time, and prospective users may balk at buying features that they believe they don't need. What should count is elapsed time to production of a new file program.

Consider a fairly typical case. From an existing file of accounts, address labels were to be printed each month. Certain special features were involved in the job (some accounts get two labels; some paired accounts get only one label between them; and so on). A COBOL program for this job grew to 4000 source cards. The Mark IV "program" for the same job took 20 minutes to write, called for punching 40 cards, and ran correctly the first time.

2. Prospective users frequently benchmark Mark IV against an existing COBOL program, and find that the Mark IV version runs slower. This is largely a comparison of apples and oranges. If the user of Mark IV does just what he did before, he is not using his new tool properly. He could, for example, perform two tasks (or more) in one pass over the file with Mark IV. Moreover, even in the original direct comparison, Mark IV is doing a great deal of checking that the COBOL program didn't do.

3. What is the message that the Mark IV salesman brings to a new prospect? No matter how well and tactfully it is disguised, the message must be "You've been doing it all wrong." If the new shop has been using 30 COBOL programmers, endlessly writing the same file programs, and the work could be done by five good people using Mark IV, then the manager has been sustaining some 25 people needlessly, perhaps for years. The installation manager, who may well see the advantages of the File Management System, has to face telling his management that there has been a mistake of some magnitude going on for years. It is only human to look around for reasons not to do that.

Currently, Mark IV and its competitors (GIS, ASI-ST, WORK 10, SCORE) are available for users of IBM 360 and 370 machines (Model 30 and up, or Model 115 and up), UNIVAC Series 70 and Series 90, and Siemens machines (of the competitive systems, only SCORE is available on machines other than IBM). What of the future? Such systems will surely be written for other brands of equipment, and for smaller machines. If a File Management System is not available for a given piece of equipment, then the user has no alternative but to write new programs for his file processing. But for an installation that deals heavily in file processing and for which a File Management System exists, it is difficult to understand why anyone would willfully choose to keep writing programs that have already been written, and written correctly.

It seems strange. Very few people write their own assemblers, and even fewer write their own COBOL processors. Why, then, do they insist on writing their own file processing programs? All the problems and intricacies of file processing have been dealt with and licked, and the work should stay done, once and for all.

Book Review

And Tomorrow...the World?--Inside IBM

by Rex Malik, Millington Ltd., 1975. 496 pages.
Distributed in the United States by the Computer
Industry Association, 1911 N. Ft. Myer Drive,
Rosslyn, Virginia 22209, \$15.

Various court actions between IBM and the Justice Department, and IBM and Control Data Corporation, put millions of company documents into the public domain, revealing some of the internal workings of one of the world's largest and richest corporations. The sifting of these documents has led to this book, supposedly the real inside story of the workings of IBM.

Nancy Foy wrote about IBM in her book The Sun Never Sets on IBM. In the review of her book in DATAMATION, April 1975, the reviewer said "Nancy Foy did her homework well and presents her findings 'without anger and after careful study.' She wrote with a sharp pen, not with a bludgeon."

Rex Malik uses a bludgeon. He also uses a strange literary style, mixing British expressions with American slang and repeating his punch lines over and over. (The phrase "dumb funny company" appears five times.) The book is badly organized and contains numerous references that are plainly wrong. IBM itself has stated that the book contains so many inaccuracies and innuendos that they cannot begin to comment on it.

And all of that would be irrelevant if the book said something. Its revelations about IBM policies and procedures are all old familiar stories to IBM-watchers. Tom Watson Senior's paternalistic rules are not current news, and even 1969 company rules on liquor and sex are out of place in such a book.

The real trouble is that Malik is an IBM-hater of the first order, who sees sinister and conspiratory actions in everything ever done within IBM.

Now, IBM's record is not like the driven snow; the company has engaged in questionable, or unethical, or downright illegal acts many times over the years. Some of these actions were perfectly proper (in the sense of being standard business practices) at the time they were made, but became questionable long after the fact, when the rules or mores of the business community changed. Also, it can be argued that most such actions are not surprising, since a company could not become large and powerful by devoting its energies to finding ways to be a good guy and helping its competitors.

But Malik sees nasty and sinister actions in everything that IBM has done. Take, for example, the provision in the 1956 consent decree that required IBM to lower its share of the market for tabulating cards:

"The case was eventually to go to the Supreme Court, where IBM lost. But the verdict did not make it essential for IBM to sell machines: it was only barred from tying the user card purchase to IBM. The loss, however, was for all practical purposes academic...IBM was allowed to devise the card specifications, and it then drew them so tight that, with its control of the best card manufacturing rotary presses in the business, no other would-be manufacturer could get in anyway."

Malik sees such actions as bad business practices. They can be looked at another way: if IBM did not react to protect its interests, the company could be indicted (say, by its stockholders) for plain stupidity. Any company is supposed to operate within the law of the land, but within that framework, its clear duty is to produce earnings and profit.

Part of the current government suit against IBM rests on the theory that excessive size, and domination of an industry is ipso facto bad. Using the precedent of the Standard Oil breakup of 65 years ago, there is an undercurrent of "Let's break up IBM." DATAMATION magazine for many months carried discussions of how that might be done, most of them carefully thought out, but no two alike. Malik joins this parade with his own version, which he modestly labels "more sensible."

It is a strange book, full of strange language, such as "Non-specialists are as likely as not to goo up the holes with chewing gum for the hell of it." Typos abound--there seems to have been no proofing of the galleys. Could Herb Grosch really have used his 701 for 170 hours a week? It is disturbing to try to cope with the curious mishmash of "facts" that are scattered through the book. For example, it seems odd to learn that IBM introduced its 600 series of calculators in 1931, and even odder to find that IBM created the EDP Division to produce the 701 after the 702 and 705 were out. The biggest surprise is this:

"It is about here that one needs to point out that if one wishes to be formally legalistic (Dr. Atanasoff and American Courts notwithstanding), all the critical working firsts in the direct ancestry of current computers originally appeared in Britain. And not only those in the direct ancestry. With the exception of the work of Zuse, and a claim in America by the late Professor Norbert Wiener (popularly known as the father of cybernetics) to have thought up the concept of the stored program computer in 1940 (though nothing came of it), the field was to be bracketed in almost every possible combination by the British. There has been much argument in Britain about who was first; the agreement is that the first operational stored program computer in the direct ancestry of today's machines was British."

Our industry is much in need of a definitive history, and a large part of that history must concern itself with IBM. But when this happy event comes about (AFIPS and the Smithsonian have been working on it for eleven years now), the actual writing had better be done by someone who understands something about computers and computing. It should probably not be either an American or a Briton, because of emotions generated by nationalism. Above all, it should be done by a historian who did not participate in the history, since human minds can so easily alter history to suit their own egos.

The first requirement (some knowledge of the field) is necessary to avoid the confusion over Who did What. Much confusion rests on the question What is a computer?, and the answer depends on What year is it? Up until 1960 or so, any device that handled numbers was called a computer; this included adding machines, punched card calculators, analog devices, and gas pumps. People called the IBM CPC a computer (even though the name says Card Programmed Calculator). Even IBM's big 1950 machine, the SSEC, was the Selective Sequence Electronic Calculator. The Burroughs company, on the other hand, was calling their E-101 (a pinboard controlled machine) a computer.

Because there is utility and virtue in distinctions (would you call a TV set with a red filter over the screen a color TV set?), our industry, around 1960, began to define "computer" along the lines of the von Neumann machine. The dividing line between computers and lesser devices was the address-modification capability; that is, the ability of a true computer to treat a word of storage as either an instruction or as data, depending on whether it headed for the decoding circuits or the accumulator. This capability is crucial, not only for ordinary address modification, but for fabricating subroutine linkages, performing second level addressing, and implementing programmed switches.

At this point in the argument, the cry is immediately heard "But we don't do such things any more by having instructions operate on other instructions!" Quite so; but we can if we wish, and computers can do it, and calculators can't. The distinction is still there, and it is useful. Those companies that make both kinds of devices (this includes IBM, Wang, Texas Instruments, and Hewlett-Packard) are careful to use the correct names. The SR-52 and SR-56 and the HP-65 and HP-67 are properly called by their makers "calculators." When those companies produce pocket computers, they will undoubtedly tell us so.

So where did the computer begin? The landmark paper by Burks, Goldstine, and von Neumann ("Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," June 28, 1946) clearly spelled out the concepts of the computer as we now know it. There have been rumors for 30 years of a memo that pre-dated that paper which showed prior invention, but that mysterious memo never appears. And if it did, what would it show? What was it that John Atanasoff produced at Iowa in 1939? What were the characteristics of Konrad Zuse's machines of the mid-30's? Did those gentlemen (giving all due credit to their brilliance and pioneering) actually invent the computer? Or did they develop binary adders and sequenced calculators? Before one lets a Federal District Court judge decide such matters, one ought to define one's terms. The original ENIAC was not a computer by today's standards, but it surely was by the terminology of its day.

We may not see a definitive history of computing in this century. When we do, it will have to be produced by someone who is aware of the shifts in terminology, who does not chalk up "B-boxes" and "index registers" as two inventions, for example, or confuse Alan Turing's proposal with the von Neumann machine.

That history will have a large intersection with the history of IBM, and its author should have a more objective view of the IBM papers than does Malik. Malik can see a conspiracy in IBM's sick leave regulations. Those parts of his book that deal with facts are obscured by irrational opinions and conjectures. He greatly overstates whatever case he tried to make, and thus wrecks all his arguments. It is conceivable that the truth about IBM is damning, but that won't be established from a base of ignorance, malice, and irrationality.

This month's contest problem is straightforward and requires very little computation.

Start with F_{150} , the 150th term of the Fibonacci sequence, defined so that F_1 , F_2 , and F_{12} are perfect squares. This is a 31-digit number that begins 996921...

The following eleven other numbers are to be subtracted from it. The square brackets indicate "greatest integer in"; the given irrational number is to be truncated at the decimal point (that is, do not round).

The resulting number is the result to be submitted as your contest entry. Most of the high precision numbers involved can be found in back issues of POPULAR COMPUTING. The 9th and 10th numbers require some modest computation.

Inasmuch as each of the 12 numbers in the problem is well defined and specific (although requiring a little digging to pinpoint), there may be many identical and correct solutions, necessitating a tiebreaker contest. There will be only one prize, our usual \$25.

All entries must be received by December 31, 1976.
Address: POPULAR COMPUTING, Box 272

Calabasas, California 91302

142
PROBLEM

CONTEST 12: COUNTDOWN

1. Factorial 29.
2. 10^9 times factorial 22.
3. 1000 times F_{119} .
4. $\left[10^{26} \text{ times the square root of } 2 \right]$.
5. $\left[10^{14} \text{ times the 20th power of } \pi \right]$.
6. $\left[3 \text{ times } 10^{21} \text{ times the natural logarithm of } 21 \right]$.
7. $\left[3 \text{ times } 10^{19} \text{ times the common logarithm of } 27 \right]$.
8. $\left[2 \text{ times } 10 \text{ times the 31st power of } \pi \right]$.
9. $\left[10^{13} \text{ times the 7th root of } 306 \right]$.
10. 10^7 times Z_{236} , where Z_{236} is the 236th term of the Z-sequence defined in issue 42.
11. $\left[10^6 \text{ times the 8th root of } 21749966 \right]$.



GIVING A SPEECH ON COMPUTING

PC43-11

1. First, some generalities. In addressing any group, technical or otherwise, there is no substitute for live rehearsal. Common courtesy dictates that the talk be planned, that it be timed (here a tape recorder can be of some help), and that it not be read. Be prepared to talk from notes or an outline. If you use audio/visual gadgetry, perhaps your slides can serve as your notes.

Plain facts about computing are startling enough so that there is no need to embellish the truth. If you offer personal opinions (and you should--that's why you're an expert) label them as such.

2. Gear your talk, as far as is possible, to your audience. Seldom is a group truly heterogeneous; some common thread brought them together. For example, a service club is likely to be made up of business and professional men and women (e.g., professions like law and medicine). If that's your audience, examples taken from advanced mathematics are going to fall flat. Perhaps examples taken from business data processing or information retrieval may appeal to such groups. Young people's groups would probably be interested in career aspects:

What makes a programmer?

What do programmers do?

What is their future?

What kind of salaries do they get?

Slightly more sophisticated groups might enjoy a simplified explanation (but remember--tell no lies!) of how a computer works. The garden club might be interested in the social implications of the computer revolution. And so it goes.

3. You can make a big impression by simply throwing out some statistics about our industry. But be careful! Today's statistics may be obsolete by next week. You should point out that the industry that centers around computers--the ultimate in precision instruments--is woefully lacking in statistical information about itself. At one time, we could make rough guesses as to the number of computers in operation; we no longer can. In fact, we cannot today even estimate the number of CPU's being manufactured each month. However, the growth rate of total computing power (measured in mips, millions of instructions per second) has held fairly steady over the years; it has doubled every 14 months or so, and shows every sign of continuing that explosive growth rate. Starting in 1955, when computing began to take off, that comes to 18 doublings, or a factor of a quarter of a million.

During the same 21-year period, raw speeds have gone from around 2000/second to perhaps 20,000,000/second; a factor of 10,000. The cost of raw computing power has gone from around 100,000 executed instructions per dollar to around a billion executed instructions per dollar, and constitutes a commodity that is freely available in any large city. (Even the pocket programmable calculators can now give you several million executed instructions per dollar.)

And so on; there is no shortage of fantastic facts about our industry. Just have these facts up to date.

4. For most non-technical people, knowledge about computers has been gained from articles in the popular press. You know how horrible some of these can be--but they have done their job. One of your jobs is to clear up some misconceptions:

No, you don't have to be a Ph.D. mathematician to make good use of a computer.

No, we can't predict the world series winner.

Yes, the machines are blinding fast.

I'll tell you whether computers can think after you tell me your definition of thinking.

Yes, I saw where a computer program had composed some poetry; wasn't that interesting?

No, computers can't solve any problem that involves numbers; there are still many problems that baffle us. Pi is known to a million places and the largest known prime number has 6002 digits--but those are extreme limits to what we have been able to do. (That largest prime is shown explicitly in issue 19.)

Clearing up misconceptions is not the same as debunking. Be a friend to the computer. If you are involved in one of the many current wars among computer people, it is best not to fight your side of it (undoubtedly the correct side) in public, at least without carefully qualifying your statements.


5. You may find your most fun in the area of blue sky dreaming. As soon as you unchain your crystal ball and peer into the future, it is clear that your views are your own. For real fun, challenge your audience to note your predictions and check up on them (they won't, but it adds a grand air of verisimilitude).

6. People may ask you for references to further reading. A canned list at this point is not of much help; it should be your list, of books or articles that you can endorse. Be ready to recommend an introductory text if someone wants it.

7. TV people have a good rule-of-thumb for making a talk move right along: "a visual a minute." This means that if you go longer than 60 seconds with nothing moving but your jaw, they'll go to sleep on you (or switch channels). This rule applies with force to the little screen, but you have much more freedom when you're there in 3D and glowing color. Nevertheless, visuals are important. Dress up your talk with action and things to look at. Make some charts, or have some slides, for example. Take along a reel of tape and some punched cards. Circuit cards from a real computer or a core plane make excellent visuals. If you can find something to hand out to be taken away (such as some punched cards), all the better. Incidentally, what should be punched on the cards, and interpreted, is something like the Lion's club motto, not your latest symbolic deck. A few hundred feet of listings from your current debugging work might be impressive, though. Or, a listing showing pairs of 8-digit numbers side by side, to illustrate the amount of arithmetic (the sum of the products) that can be done by a fast machine in one second (at 60 lines per page, that would be over 4000 pages).

8. After recounting the glories of the computer world, and telling how frightfully smart we are, you might consider pointing out that we're still groping in the dark. There are lots of problems left to solve (both problems of the world in general and problems about how to use computers effectively) and we still have an acute shortage of brains. This goes particularly well with young people. They'll feel more comfortable knowing that it's still the ground floor, and will be for quite a few more years. Our industry is settling down somewhat, but it's still exciting and has many pioneering aspects. It can be anything but dull, for the right type of person. And the excitement never seems to fade

9. Close with some positive thoughts. The world's chess champion may someday be a computer program. We may eventually simplify the country's paperwork, or even eliminate most of it. Computers will be part of all our lives, and will be in most homes and cars. The computer is a powerful tool and will probably help us to solve many of our urgent problems.



Contest 7 Results

The problem presented in Contest 7 was the following.

Start with the natural numbers. At level 1, take one number, reject one number, take one, reject one, indefinitely. Level 1 thus yields the odd integers. At level 2, take the output from level 1: take two numbers, reject two numbers, take two, reject two, indefinitely. Repeat this process: at level K, take K numbers, reject K numbers, ... What numbers will survive this sieving process?

The winning solution, unquestionably, comes from Tom Duff and Hugh Redelmeier, graduate students in computer science at the University of Toronto, who submitted 1199 terms of the resulting sequence. The first 58 of these terms, and some larger ones, are shown. Mr. Duff writes:

"We notice first that if we want only the first N numbers that the sieve generates, we need only simulate the first N-1 levels of the sieve, since the levels from N onward will all let at least the first N numbers pass. Each level of the K-level sieve may be regarded as a co-routine or asynchronous process which is in a producer-consumer relationship with the levels immediately above and below it. Level 1 of the sieve may be represented by the following piece of programme:

```

Loop:
    for J:=1 until I do
        receive M from level I-1
        send M to level I+1
    end
    for J:=1 until I do
        receive M from level I-1
    end
    goto Loop

```

A whole set of these co-routines, running in parallel, act a little like a bucket brigade. When a member of the bucket brigade receives a number from the previous member, it either passes the number on to the next member, or it discards the number. Of course, the first and last levels of the sieve are a little different, since the first level has no previous level to receive numbers from, and the last one prints the numbers out instead of passing them on.

With a little effort, this algorithm can be programmed to be relatively quick, but no matter how hard you try, its speed will still be at best a linear function of the result. Since the numbers outputted by the K-level sieve increase approximately exponentially, there is little hope of producing significant output of the sieve with any programme that has to simulate the sieve's operation.

1		1
2		3
3		9
4		25
5		57
6		145
7		337
8		793
9		1921
10		3849
11		8835
12		18889
13		41473
14		92305
15		203211
16		432699
17		944313
18		2027529
19		4077769
20		8745153
21		18133305
22		37898113
23		80713737
24		169730259
25		358760457
26		750591867
27		1575313473
28		3255787851
29		6751959507
30		14108682265
31		29364255033
32		61173205587
33		126792880201
34		261786645129
35		542760030745
36		1090481316033
37		2254104779211
38		4576926103825
39		9375021033745
40		19362713813451
41		39746772236619
42		81889654169481
43		168348435476475
44		346352979792385
45		710707885619259
46		1454261791467673
47		2985844385599497
48		6112722593126091
49		12525096166152969
50		25453257177612675
51		52120032930644041
52		106216796997067065
53		217388980361779977
54		443667202522163353
55		905860955252113281
56		1851430774702624849
57		3778065321123848833
58		7700902493000163393
	100	59721045190927381881499710261123
	500	8064434252147229723515492934230308242133559888901 3727097631334253455067996456969941417811642485116 1734818267880537671943571900003441962106674037047 407233
	1000	53135325489226165408921584229151796140386264683541 4010630692427060681617155340580221063014675818895 1608129661264218913555204037441755816463307268532 6457016253865930987542603153162451709294642721995 3290391722638945523574345904078467012319750764268 7546266981530584938038605859667128033923096441412 4937911825
	1199	5119095356622829848295226812411528502341478017175 9639533078954241574659785529231843938218888817594 2482021288456532073349170622212188875613312551219 7445952463318340192492856985443353429511600258319 5017528328736772239016794711566448097482403123988 7108383447448346850596669456669456665669177556178 3991820852913232896296356290846110437322072526673 577069709097224927441932353

Some results of the
K-level Sieve Problem

Let us analyze the behaviour of this programme. Each member of the bucket brigade receives a certain number of numbers from the previous member and passes along a certain fraction of them to the next member. When level I passes along its Nth number to level I+1, it will have received

$$2*N - ((N-1) \bmod I) - 1$$

numbers from level I-1. The Kth number outputted by the sieve is just the length of the sequence of numbers that must be inputted to the first level of the sieve to get K numbers out of the Kth level. We can compute that number by iterating on the above formula, like this:

```
N:=K
For I:=K-1 step -1 to 1 do
    N:=2*N-((N-1) mod I)-1
end
```

Thus, a complete programme to print the first KMAX numbers generated by the K-level sieve is:

```
for K:=1 until KMAX do
    N:=K
    for I:=K-1 step -1 until 1 do
        N:=2*N-((N-1) mod I)-1
    end
    print K, N
end
```

In order to simplify the calculations in the inner loop, the above programme can be transformed by replacing K and N with K+1 and N+1. This yields the following programme:

```
for K:=0 until KMAX-1 do
    N:=K
    for I:=K step -1 until 1 do
        N:=N*2-N mod I
    end
    print K+1, N+1
end
```

Our programme implements the last algorithm given above in UNIX assembly language. The main problem is that the variable N must be stored to extremely high precision. Our programme represents N as a sequence of base 10000 digits, each of which is stored in a single PDP-11 word."

We think that Duff and Redelmeier's work is a superb piece of computing. Members of our staff wrote eight different programs for the K-level sieve, the best of which yielded 42 terms of the sequence. The problem appears to be intrinsically useless, but is still a splendid exercise in logical analysis that calls for a high level of sophistication to achieve significant output. With 1199 terms of the final sequence now known, the problem passes into the realm of outstanding textbook problems.

Book Review

PC43-17

Introduction to Computer Applications for Non-science Students

by William Ralph Bennett, Jr.

Prentice-Hall, 1976, 207 pages 8 1/2 x 11.

The title of this book suggests a low-level elementary text on computing intended for a survey course. This is misleading. The book assumes a high level of sophistication and extensive exposure to computers. Its language for computation is BASIC, and it also assumes that the reader has carefully studied the BASIC reference manual for his installation. The author plunges into the "applications" directly, emphasizing scientific, engineering, and statistical applications. He has apparently done considerable computing, but all in BASIC and all as an amateur. A reader who is not familiar with mathematical and scientific notation will get lost rather quickly.

The advanced features of BASIC are introduced in context and casually. For example, the reader will find himself in a discussion of loops (of the simplest type), followed by FOR...NEXT control, followed by nested loops, all within two pages.

Bennett leaps from sophistication to naivete and back. He describes the use of Hilbert matrices as a test tool, for example (which is a rather advanced idea) and two pages later advises his readers not to use Simpson's Rule for integrating (simply use the Trapezoidal Rule and take more intervals)--this is naive, inasmuch as Simpson's Rule takes about the same amount of code and CPU time per interval, but with far more efficiency.

The book is full of curious indications of the author's computational immaturity. The statement "There are many important functions (e.g., SIN, COS, EXP) which can only be computed by the use of power-series expansions in the independent variable" is fatuous (probably most calculations of those functions today do not use power-series expansions). The pocket calculators, for example, do not. Speaking of pocket calculators, the author calls the HP-65 a computer, and goes on to show that he knows very little about such machines. In the same vein, the 10th problem in the book suggests a way to find prime numbers that is quite inefficient, and the author graciously credits his daughter with the method of solution. (Since the problem calls only for primes up to 1000, little damage would be done by the stated inefficiency, except that students will carry it up to ten million and waste hours of CPU time. Besides, it is not


the function of a book on computing to foster bad methods. Or, it shouldn't be.)

Bennett advocates plunging into coding on any problem; flowcharts are a "needless encumbrance." (In all fairness, he does point out that "It pays to give a great deal of advance thought to difficult problems before going 'on line'.")

The author's literary style includes pedantic words like "extremum" but seldom gets the placement of the word "only" correct. ("It only requires programming facility...")

Nearly half the book is concerned with computer simulation of the problem of the monkeys typing at random to reproduce the books in the British Museum, and with elementary cryptography. On the monkey problem, the author has evidently done much work and speaks with authority (indeed, this section may be a real contribution to the literature). In the field of cryptography, the level is about that of Yardley's The American Black Chamber (1931).

This is one of those books that could set the art of computing back a decade or so. It purports to teach use of the computer, but fosters bad computing at every step. The author's prestigious position (Charles Baldwin Sawyer Professor of Engineering and Applied Science at Yale) suggests the imprimatur of authority, whereas the reader is being exposed to an amateur and dilettante who has enjoyed playing with BASIC on a minicomputer. If such books cannot be written by knowledgeable people, at least they could be proofed by someone who understands computing. The rebuttal, I suppose, is that computing is so cheap these days that it makes little difference whether one computes efficiently or inefficiently. Perhaps. But condoning sloppy work is one thing; teaching it systematically is another. As Joe Weizenbaum has so neatly put it, "Whatever it is that hospitals do, they shouldn't spread disease." One might consider the effects of this book being used by an instructor who knows even less about computing than its author. It is no wonder that industry has a low opinion of the output of our colleges in the computing field.



A New Random Number Generator

PC43-19

In the program "Sequences" of Dr. Mordecai Schwartz (PC42-3), the following recursion is used as a random number generator:

$$\text{new} = \text{fractional part of } (\text{old} + \pi)^5$$

This generator was tested by Associate Editor David Babcock, with the following results.

The distribution of the leading digits of each new number (the frequency test):

0	1	2	3	4	5	6	7	8	9
988	976	1027	998	1008	1012	926	1021	987	1057

which gives a chi-squared value of 6.296 for a p of .72.

The leading two digits show the following distribution (serial test):

	0	1	2	3	4	5	6	7	8	9
0	85	105	108	88	86	90	110	111	110	95
1	96	89	94	104	115	101	88	112	92	85
2	100	90	110	109	111	105	99	103	107	93
3	107	90	101	94	105	110	87	108	98	98
4	93	93	94	122	98	86	102	117	99	104
5	108	91	114	102	80	117	95	108	100	97
6	77	85	102	103	87	118	90	73	96	95
7	107	105	89	106	124	119	89	95	94	93
8	90	92	99	118	107	69	86	114	103	109
9	93	106	96	92	106	100	121	125	114	104

The chi-squared value on this data is 83.44, which yields a probability of less than .01.

Using the entire generated number, the D^2 test is as follows. Two successive random numbers are taken as the coordinates of a point in the unit square. For two such points, the square of the distance between them is calculated:

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 = D^2$$

These values of D^2 are distributed as follows:

.0	.1	22702	22702	23483
.1	.2	17029	39731	40981
.2	.3	14687	54418	54930
.3	.4	11598	66016	66202
.4	.5	10062	76078	75299
.5	.6	6812	82890	82560
.6	.7	5725	88615	88235
.7	.8	4142	92757	92516
.8	.9	3239	95996	95559
.9	1.0	1593	97589	97493
1.0	1.1	967	98556	98570
1.1	1.2	550	99106	99205
1.2	1.3	481	99587	99579
1.3	1.4	344	99931	99793
1.4	1.5	0	99931	99908
1.5	1.6	69	100000	99965
1.6	1.7	0	100000	99990
1.7	1.8	0	100000	99998
1.8	1.9	0	100000	99999
1.9	2.0	0	100000	100000

from this value ↑
 up to but not including this value of D^2 ↑
 observed frequency ↑
 cumulative totals ↗
 theoretical frequencies ↑